

# Sandwich Batch Normalization: A Drop-In Replacement for Feature Distribution Heterogeneity

Xinyu Gong Wuyang Chen Tianlong Chen Zhangyang Wang

Department of Electrical and Computer Engineering, the University of Texas at Austin

{xinyu.gong, wuyang.chen, tianlong.chen, atlaswang}@utexas.edu

## Abstract

We present *Sandwich Batch Normalization (SaBN)*, a frustratingly easy improvement of Batch Normalization (BN) with only a few lines of code changes. SaBN is motivated by addressing the inherent feature distribution heterogeneity that one can be identified in many tasks, which can arise from data heterogeneity (multiple input domains) or model heterogeneity (dynamic architectures, model conditioning, etc.). Our SaBN factorizes the BN affine layer into one shared sandwich affine layer, cascaded by several parallel independent affine layers. Concrete analysis reveals that, during optimization, SaBN promotes balanced gradient norms while still preserving diverse gradient directions – a property that many application tasks seem to favor. We demonstrate the prevailing effectiveness of SaBN as a drop-in replacement in four tasks: conditional image generation, neural architecture search (NAS), adversarial training, and arbitrary style transfer. Leveraging SaBN immediately achieves better Inception Score and FID on CIFAR-10 and ImageNet conditional image generation with three state-of-the-art GANs; boosts the performance of a state-of-the-art weight-sharing NAS algorithm significantly on NAS-Bench-201; substantially improves the robust and standard accuracies for adversarial defense; and produces superior arbitrary stylized results. We also provide visualizations and analysis to help understand why SaBN works. Codes are available at: <https://github.com/VITA-Group/Sandwich-Batch-Normalization>.

## 1. Introduction

This paper presents a simple, light-weight, and easy-to-implement modification of Batch Normalization (BN) [30], motivated by various observations [69, 13, 61, 62] drawn from several applications, that *BN has troubles standardizing hidden features with a heterogeneous, multi-modal distribution*. We call this phenomenon *feature distribution heterogeneity*. Such heterogeneity of hidden features could arise from multiple causes, often application-dependent:

- One straightforward cause is the input *data heterogeneity*. For example, when training a deep network on a diverse set of visual domains, that possess significantly different statistics, BN is found to be ineffective in normalizing the activations with only a single mean and variance [13], and often needs to be re-set or adapted [40].
- Another intrinsic cause could arise from the *model heterogeneity*, i.e., when the training is, or could be equivalently viewed as, on a set of different models. For instance, in neural architecture search (NAS) using weight sharing [43, 17], training the supernet during the search phase could be considered as training a large set of sub-models (with many overlapped weights) simultaneously. As another example, for conditional image generation [47], the generative model could be treated as a set of category-specific sub-models packed together, one of which would be “activated” by the conditional input each time.

The vanilla BN (Figure 1 (a)) fails to perform well when there is data or model heterogeneity. Recent trends split the affine layer into multiple ones and leverage input signals to modulate or select among them [12, 13] (Figure 1 (b)); or even utilize several independent BNs to address such disparity [69, 61, 62, 67]. While those relaxations alleviate the data or model heterogeneity, we suggest that they might be “*too loose*” in the normalization or regularization effects.

Let us take the conditional image generation task of GANs as a concrete motivating example to illustrate our rationale. A GAN model is trained with an image dataset containing various image classes, tending to capture the distribution of real samples to produce similar image examples. As a helpful remedy for the generator to encode class-specific information, Categorical Conditional Batch Normalization (CCBN), is widely used in the conditional image generation [5, 47, 48]. It is composed of a normalization layer and a number of following separate affine layers for different image classes, which allows class-specific information to be encoded separately, thus enables the gen-

erator to generate more vivid examples.

But what might be missing? Unfortunately, using separate affines ignores one important fact that different image classes, while being different, are **not totally independent**. Considering that images from the same dataset share some common characteristic (e.g., the object-centric bias for CIFAR images), it is convincing to hypothesize the different classes to be largely overlapped at least (i.e., they still share some hidden features despite the different statistics). To put it simply: while it is oversimplified to normalize the different classes as “the same one”, it is also unfair and unnecessary to treat them as “totally disparate”.

More application examples can be found that all share this important structural feature prior. [43, 17, 67] train a large variety of child models, constituting model heterogeneity; but most child architectures inevitably have many weights in common since they are sampled from the same supernet. Similarly, in adversarial training, the model is trained by a mixture of the original training set (“*clean examples*”) and its attacked counterpart with some small perturbations applied (“*adversarial examples*”). But the clean examples and adversarial examples could be largely overlapped, considering that all adversarial images are generated by perturbing clean counterparts only minimally.

**Our Contributions:** Recognizing the need to address feature normalization with “harmony in diversity”, we propose a new **SaBN** as illustrated in Fig 1 (c). SaBN modifies BN in an embarrassingly simple way: it is equipped with two cascaded affine layers: a shared unconditional *sandwich affine* layer, followed by a set of independent affine layers that can be conditioned. Compared to CCBN, the new sandwich affine layer is designed to inject an inductive bias, that all re-scaling transformations will have a shared factor, indicating the commodity.

We then dive into a detailed analysis of why SaBN shows to be effective, and illustrate that during optimization, SaBN promotes **balanced gradient norms** (leading to more fair learning paces among heterogeneous classes and features), while still preserving **diverse gradient directions** (leading to each class leaning towards discriminative feature clusters): a favorable inductive bias by many applications.

Experiments on the applications of conditional image generation and NAS demonstrate that SaBN addresses the *model heterogeneity* issue elegantly, improving generation quality in GAN and the search performance in NAS in a plug-and-play fashion. To better address the *data heterogeneity* altogether, SaBN could further integrate the idea of split/auxiliary BNs [69, 61, 62, 67], to decompose the normalization layer into multiple parallel ones. That yields the new variant called **SaAuxBN**, demonstrated by the example of adversarial training. Lastly, we extend the idea of SaBN to Adaptive Instance Normalization (AdaIN) [28] and show the resulting **SaAdaIN** to improve arbitrary style transfer.

## 2. Related Work

### 2.1. Normalization in Deep Learning

Batch Normalization (BN) [30] made critical contributions to training deep convolutional networks and nowadays becomes a cornerstone of the latter for numerous tasks. BN normalizes the input mini-batch of samples by the mean and variance, and then re-scales them with learnable affine parameters. The success of BNs was initially attributed to overcoming internal covariate shift [30], but later on raises many open discussions on its effect of improving landscape smoothness [55]; enabling larger learning rates [4] and reducing gradient sensitivity [2]; preserving the rank of pre-activation weight matrices [11]; decoupling feature length and direction [35]; capturing domain-specific artifacts [40]; reducing BN’s dependency on batch size [29, 56]; preventing elimination singularities [53]; and even characterizing an important portion of network expressivity [20].

Inspired by BN, a number of task-specific modifications exploit different normalization axes, such as Instance Normalization (IN) [57] for style transfer; Layer Normalization (LN) [3] for recurrent networks; Group Normalization (GN) [60] for tackling small batch sizes; StochNorm [36] for fine-tuning; Passport-aware Normalization [72] for model IP protection; and [38, 58, 73] for image generation.

Several normalization variants have been proposed by modulating BN parameters, mostly the affine layer (mean and variance), to improve the controlling flexibility for more sophisticated usages. For example, Harm *et al.* [12] presented Conditional BN, whose affine parameters are generated as a function of the input. Similarly, Conditional IN [19] assigned each style with independent IN affine parameters. In [47], the authors developed Categorical Conditional BN for conditional GAN image generation, where each generated class has its independent affine parameters. Huang & Belongie [28] presented Adaptive IN (AdaIN), which used the mean and variance of style image to replace the original affine parameter, achieving arbitrary style transfer. Spatial adaptivity [51] and channel attention [39] managed to modulate BN with higher complexities.

A few latest works investigate to use multiple normalization layers instead of one in BN. [13] developed mode normalization by employing a mixture-of-experts to separate incoming data into several modes and separately normalizing each mode. [69] used two separate BNs to address the domain shift between labeled and unlabeled data in semi-supervised learning. Very recently, [62, 61] revealed the two-domain issue in adversarial training and find improvements by using two separate BNs (AuxBN).

### 2.2. Brief Backgrounds for Related Applications

We use four important applications as testbeds. All of them appear to be oversimplified by using the vanilla BN,

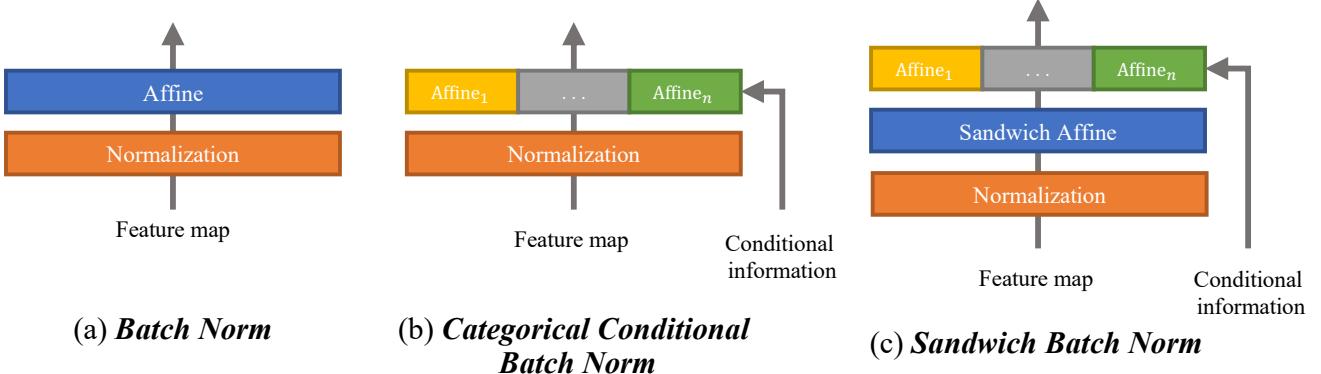


Figure 1. Illustration of (a) vanilla batch normalization (**BN**), composed of one normalization layer and one affine layer; (b) **Categorical Conditional BN**, composed of one normalization layer following a set of independent affine layers to intake conditional information; (c) **Sandwich BN**, sequentially composed of one normalization layer, one shared sandwich affine layer, and a set of independent affine layers.

where the feature homogeneity and heterogeneity are not properly handled. We briefly introduce them below, and will concretely illustrate where the heterogeneity comes from and how our methods deal with it in Sec. 3.

**Generative Adversarial Network** GAN has been prevailing since its origin [23] for image generation. Many efforts have been made to improve GANs, such as modifying loss function [1, 25, 32], improving network architecture [71, 34, 22, 6] and adjusting training procedure [33]. Recent works also tried to improve the generated image quality by proposing new normalization modules, such as Categorical Conditional BN and spectral normalization [47].

**Neural Architecture Search (NAS)** The goal of NAS is to automatically search for an optimal model architecture for the given task and dataset. It was first proposed in [74] where a reinforcement learning algorithm iteratively samples, trains and evaluates candidate models from the search space. Due to its prohibitive time cost, the weight-sharing mechanism was introduced [52] and becomes a popular strategy [43]. However, weight-sharing causes performance deterioration due to unfair training [10], motivating other alternatives to accelerate NAS [65, 7, 45]. Besides, a few NAS benchmarks [66, 18, 70] were recently released, with ground-truth accuracy for candidate models pre-recorded, enabling researchers to evaluate the performance of the search method more easily [64, 8, 59].

**Adversarial Robustness** Deep networks are notorious for the vulnerability to adversarial attacks [24]. In order to enhance adversarial robustness, countless defense approaches have been proposed [15, 50, 63, 46, 41, 44, 16]. Among them, adversarial training (AT) [44] is arguably the strongest, which trains the model over a mixture of clean and perturbed data. The normalization in AT had not been

studied in-depth until the pioneering work [61] introduced an auxiliary batch norm (AuxBN) to improve the clean image recognition accuracy.

**Neural Style Transfer** Style transfer [21] generates a stylized image, by combining the content of one image with the style of another. Various improvements are made on the normalization methods [9]. [57] proposed Instance Normalization (IN), improving the stylized quality of generated images. Conditional Instance Normalization (CIN) [19] and Adaptive Instance Normalization (AdaIN) [28] enable a single network to perform multiple/arbitrary style transfer.

### 3. Sandwich Batch Normalization

Given the input feature  $\mathbf{x} \in \mathbb{R}^{N \times C \times H \times W}$  ( $N$  denotes the batch size,  $C$  the channel number,  $H$  height and  $W$  width), the vanilla **Batch Normalization (BN)** works as:

$$\mathbf{h} = \gamma \left( \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \right) + \beta, \quad (1)$$

where  $\mu(\mathbf{x})$  and  $\sigma(\mathbf{x})$  are the running estimates (or batch statistics) of input  $\mathbf{x}$ 's mean and variance along ( $N, H, W$ ) dimensions.  $\gamma$  and  $\beta$  are the learnable parameters of the affine layer, and both are of shape  $C$ . However, the vanilla BN only has a single re-scaling transform and will treat any latent feature from one single distribution.

As an improved variant, **Categorical Conditional BN (CCBN)** [47] is proposed to remedy the heterogeneity issue in the task of conditional image generation, boosting the quality of generated images. CCBN has a set of independent affine layers, whose activation is conditioned by the input domain index and each affine layer is learned to capture the class-specific information. It can be expressed as:

$$\mathbf{h} = \gamma_i \left( \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \right) + \beta_i, i = 1, \dots, C, \quad (2)$$

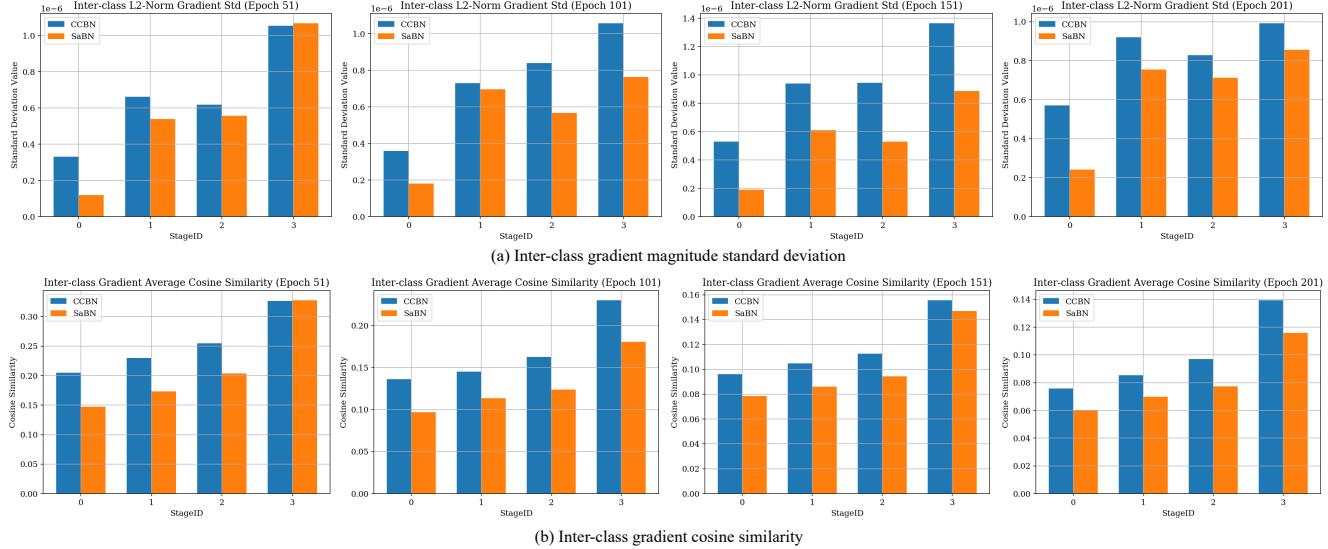


Figure 2. **The visualization of standard deviations of gradient magnitudes** (the lower the better, i.e., more balanced optimization paces) and cosine similarity across different classes (the lower the better, i.e., more diverse features learned). The x-axis of each plot denotes the depth of generator network, where each generator is composed of four stages of convolution blocks.

where  $\gamma_i$  and  $\beta_i$  are parameters of the  $i$ -th affine layer. Concretely,  $i$  is the expected output class in the image generation task [48]. However, we argue that this “separate/split” modification might cause imbalanced learning for different classes. Since the training data from each class might vary a lot (different number of examples, complicated/simple textures, large/small inner-class variation, etc.), different individual affine layers might have significantly diverged convergence speed, impeding the proper training of the whole network. Dominant classes will introduce stronger inductive biases on convolutional layers than minor classes.

To better handle the imbalance, we present **Sandwich BN (SaBN)**, that is equipped with a shared sandwich affine layer and a set of independent affine layers:

$$\mathbf{h} = \gamma_{sa}(\frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}) + \beta_{sa} + \beta_i, i = 1, \dots, C. \quad (3)$$

As depicted in Fig. 1 (d),  $\gamma_{sa}$  and  $\beta_{sa}$  denote the new sandwich affine layer, while  $\gamma_i$  and  $\beta_i$  are the  $i$ -th affine parameters, conditioned on categorical inputs. Implementation-wise, SaBN only takes **a few lines of code changes** over BN: please see the **supplement** for pseudo codes.

### 3.1. Why SaBN meaningfully works?

One might be curious about the effectiveness of SaBN, since at the inference time, the shared sandwich affine layer can be multiplied/merged into the independent affine layers, making the inference form of SaBN completely identical to CCBN. So where is its real advantage?

By the analysis below, we argue that: SaBN provides *a favorable inductive bias for optimization*. During training,

we observe that SaBN promotes **balanced gradient norms** (leading to more fair learning paces among heterogeneous classes and features), while still preserving **diverse gradient directions** (leading to each class leaning towards discriminative feature clusters).

We take the training of the conditional image generation task as an example. As one of the state-of-the-art GANs, SNGAN [47] successfully generates high-quality images in the conditional image generation task with CCBN. Intuitively, it uses independent affine layers to disentangle the image generation of different classes. We consider analyzing the following two models: 1) SNGAN (equipped with CCBN originally); 2) SNGAN-SaBN (simply replaces CCBN with our SaBN). Both of them are trained on the same subset of ImageNet, with the same training recipe (see Sec. 4.1 for details). Here we analyze the difference of inter-class gradients of the generator in two aspects: (1) gradient magnitude and (2) gradient direction.

#### SaBN Encourages Balanced Gradient Magnitudes.

For the first aspect, we analyze the standard deviation of the  $l_2$ -norm (magnitude) of generator’s gradients among different classes in Fig. 2 (a). Concretely, we take the gradients from weights of convolution layers, which are right before the normalization modules. We observe that the gradient norms from SNGAN-SaBN mostly have lower standard deviations, indicating that the gradient magnitudes of different classes in SNGAN-SaBN are more balanced. A balanced distribution of gradient magnitudes is found to be preferred for model training, avoiding some features dominating the others, and facilitating the optimization of all sub-tasks at similar paces [68].

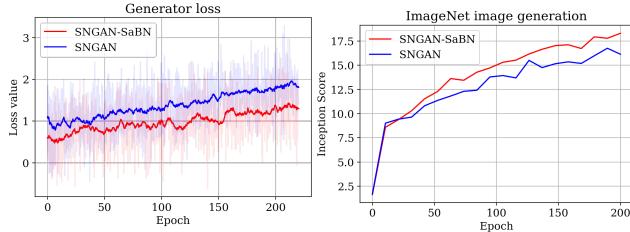


Figure 3. (left) The generator loss  $L_G$  of SNGAN and SNGAN-SaBN during training phase (ImageNet). (right) The Inception Score curves during training.

**SaBN Preserves Diversity of Gradient Directions.** We then visualize the averaged cosine similarity of generator’s gradients from different classes during training in Fig. 2 (b). Specifically, we define the inter-class gradient similarity  $g_{\text{inter}}$ , which aims to measure the divergence of gradients from different input class labels  $y$  and averaged over latent vectors ( $z$ ):

$$g_{\text{inter}}^l = \frac{1}{m} \sum_{i=0}^{m-1} \mathcal{C} \left( \nabla_{\theta^l} \mathcal{L}(G(z_i, y_j)) \Big|_{j=0}^{n-1} \right) \quad (4)$$

Here the generator is denoted by  $G$ , and  $\nabla_{\theta^l} \mathcal{L}(G(z_i, y_j))$  represents the gradients on convolution layers of the  $l$ -th stage of the generator (i.e., the derivative of loss  $\mathcal{L}$  with respect to parameters in  $l$ -th stage  $\theta^l$ ; we omit the discriminator here).  $m$  and  $n$  are the total number of latent vectors  $z$  and class labels  $y$ , respectively. Function  $\mathcal{C}$  calculates the averaged pair-wise cosine similarity of inputs:

$$\mathcal{C}(\mathbf{v}_i |_{i=1}^N) = \frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j=1, j \neq i}^N \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}. \quad (5)$$

We can see that SNGAN-SaBN has lower  $g_{\text{inter}}$ , indicating the gradients from different classes are more diverse in their directions. This enables the generator to capture richer discriminative information among different classes and contribute to a visually more diverse generation.

**Summary:** The above two characteristics brought by SaBN, i.e., overall more balanced gradient norms, and inter-class more diverse gradient directions, together draw the big picture why SaBN facilitates the optimization especially in the presence of heterogeneous or multi-domain features. Fig. 3 visualizes GAN training with and without SaBN: SNGAN with SaBN can achieve much lower generator loss  $L_G$  (Eq. 6) than the original SNGAN (Fig. 3 left), and the generation quality of the former consistently outperforms the latter (by Inception Score [54], Fig. 3 right).

## 4. Experiments

Sandwich Batch Normalization is an effective plug-and-play module. In this section, we present the experiment re-

Table 1. Best **Inception Scores** (“IS”,  $\uparrow$ ) and **FIDs** ( $\downarrow$ ) achieved by conditional SNGAN, BigGAN, and AutoGAN-top1, using CCBN and SaBN on CIFAR-10 and ImageNet (dogs & cats).

Model	CIFAR-10		ImageNet (dogs & cats)	
	IS	FID	IS	FID
AutoGAN-top1	8.43	10.51	-	-
BigGAN	8.91 <sup>1</sup>	8.57 <sup>1</sup>	-	-
SNGAN	8.76	10.18	16.75	79.14
AutoGAN-top1-SaBN	8.72(+0.29)	9.11(-1.40)	-	-
BigGAN-SaBN	9.01(+0.10)	8.03(-0.54)	-	-
SNGAN-SaBN	8.89(+0.13)	8.97(-1.21)	18.31(+1.56)	60.38(-18.76)

sults of naively applying it into two different tasks: conditional image generation and neural architecture search.

### 4.1. Conditional Image Generation with SaBN

Following the discussion in the previous section, we present detailed settings and main results on the conditional image generation task using SaBN in this section. We choose three representative GAN models, SNGAN, BigGAN [5] and AutoGAN-top1 [22], as our baselines. The generator of SNGAN and BigGAN are equipped with CCBN originally. AutoGAN-top1 does not have any normalization layer and is designed for unconditional image generation, thus we manually insert CCBN into its generator to adapt it to the conditional image generation task. We then construct SNGAN-SaBN, BigGAN-SaBN, and AutoGAN-top1-SaBN, by simply replacing all CCBN in the above baselines with our proposed SaBN.

GAN models in our paper are all trained with the hinge version adversarial loss [47, 5]:

$$L_D = -\mathbb{E}_{(x,y) \sim p_{\text{data}}} [\min(0, -1 + D(x, y))] - \mathbb{E}_{z \sim p_z, y \sim p_{\text{data}}} [\min(0, -1 - D(G(z, y), y))], \quad (6)$$

$$L_G = -\mathbb{E}_{z \sim p_z, y \sim p_{\text{data}}} D(G(z, y), y),$$

where  $L_D$  and  $L_G$  denote discriminator loss and generator loss respectively.

We test all the above models on CIFAR-10 dataset [37] (10 categories, resolution  $32 \times 32$ ). Furthermore, we test SNGAN and SNGAN-SaBN on high-resolution conditional image generation task with ImageNet [14], using the subset of all 143 classes belonging to the dog and cat super-classes, cropped to resolution  $128 \times 128$  following [47]’s setting. Inception Score [54] (the higher the better) and FID [27] (the lower the better) are adopted as evaluation metrics. We summarize the best performance the models have achieved during training into Table 1. We find that SaBN can consistently boost the generative quality of all three baseline GAN models, which demonstrates the effectiveness of the injected shared sandwich affine layer. We also provide visualization results of generated images in Fig. 4. Since images of CIFAR-10 dataset [37] are too small to tell difference,

<sup>1</sup>Results obtained by using the author’s officially unofficial PyTorch BigGAN implementation.

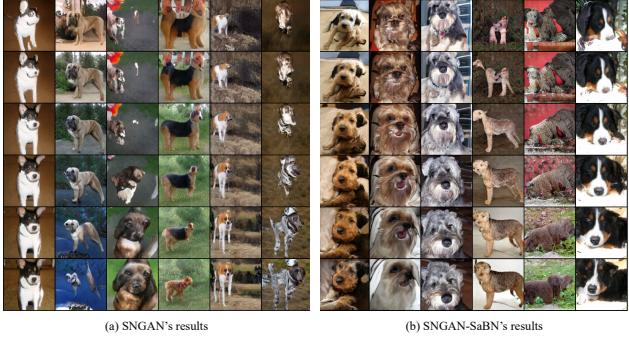


Figure 4. The image generation results of SNGAN and SNGAN-SaBN on ImageNet. Each column is corresponding to a specific image class. Images are chosen without cherry-pick.

we only visualize the results on ImageNet [14]. Specifically, we compare the generation results of SNGAN and SNGAN-SaBN. The images generated by SNGAN-SaBN are more visual appealing, showing better quality.

#### 4.2. Architecture Heterogeneity in Neural Architecture Search (NAS)

Recent NAS works formulate the search space as a weight-sharing supernet that contains all candidate operations and architectures, and the goal is to find a sub-network that of the optimal performance. As one of the representative works in NAS, DARTS [43] solves the search problem by assigning each candidate operation a trainable architecture parameter  $\alpha$ . Model weights  $\omega$  and architecture parameters  $\alpha$  are optimized to minimize the cross-entropy loss in an alternative fashion. After searching, the final architecture is derived by choosing the operation with the highest  $\alpha$  value on each edge.

However, such formulation introduces a strong model heterogeneity. As shown in Fig. 5, the output of each layer is the sum of all operations' output, weighted by associated architecture parameters  $\alpha$ . Such mixed model heterogeneity could be harmful to the search, making the algorithm hard to distinguish the contribution of each operation. Inspired by the application of CCBN in GANs, we preliminarily attempt to use CCBN disentangling the mixed model heterogeneity from the previous layer, by replacing the BN in each operation path with a CCBN (namely DARTS-CCBN). The total number of affine paths is equal to the number of candidate operations in the previous layer, and the conditional index  $i$  of CCBN is obtained by applying a multinomial sampling on the softmax of previous layers' architecture parameters (shown in Fig. 5). The search results of the vanilla DARTS and DARTS-CCBN are reported in Tab. 2. Compared with vanilla DARTS, DARTS-CCBN does not show consistent improvement w.r.t. the search results. We argue that the brutal “separate/split” modification

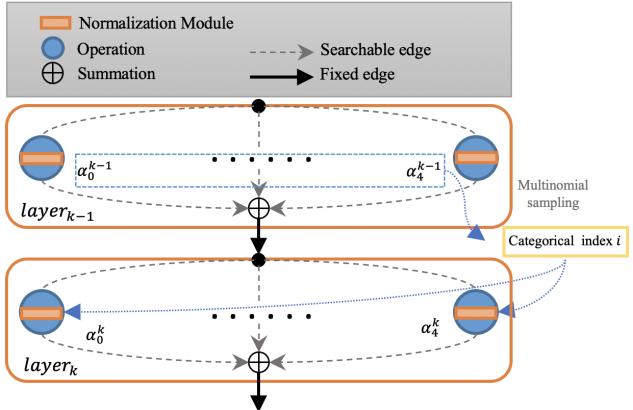


Figure 5. Two consecutive layers in the supernet. By default, a BN is integrated into each parameterized operation in vanilla DARTS. The output of each layer is the sum of all operation paths' output, weighted by their associated architecture parameter  $\alpha$ .

Table 2. The ground-truth top-1 accuracy of the final searched architecture on NAS-Bench-201. DARTS-SaBN achieves the highest accuracy, with the lowest standard deviation. Bench optimal denotes the best test accuracy achievable in NAS-Bench-201.

Method	CIFAR-100	ImageNet16-120
DARTS	$44.05 \pm 7.47$	$36.47 \pm 7.06$
DARTS-affine	$63.46 \pm 2.41$	$37.26 \pm 7.65$
DARTS-CCBN	$62.16 \pm 2.62$	$31.25 \pm 6.20$
DARTS-SaBN (ours)	<b><math>71.56 \pm 1.39</math></b>	<b><math>45.85 \pm 0.72</math></b>
Bench Optimal	73.51	47.31

in CCBN might cause imbalanced learning for different operations due to their intrinsic difference, therefore leading to unfair competition among candidate operations.

To better handle such an unbalanced issue, we consider using SaBN instead of CCBN (DARTS-SaBN). The injected shared sandwich affine layer is designed to balance the learning among different operations, imposing a more fair competition among candidate operations. As shown in Tab. 2, we can observe that DARTS-SaBN outperforms the vanilla DARTS and DARTS-CCBN significantly, whose performance is even close to the Bench optimal. The performance gap between DARTS-CCBN and DARTS-SaBN demonstrates the effectiveness of the sandwich affine layer. We further include an additional ablation variant DARTS-affine, which simply enables the affine layer of the BN in DARTS. DARTS-SaBN also outperforms DARTS-affine with a considerable margin, implying the independent conditional affine layers are also important.

## 5. Extended Applications of Sandwich Batch Normalization

In this section, we explore the possibility to extend Sandwich Batch Normalization to more tasks with minor modifications. Concretely, we apply two variants of SaBN on adversarial robustness and arbitrary style transfer.

### 5.1. Sandwich Auxiliary Batch Norm (SaAuxBN) in Adversarial Robustness

AdvProp [61] successfully utilized adversarial examples to boost network Standard Testing Accuracy (SA) by introducing Auxiliary Batch Norm (AuxBN). The design is quite simple: an additional BN is added in parallel to the original BN, where the original BN (clean branch) takes the clean image as input, while the additional BN (adversarial branch) is fed with only adversarial examples during training. That intuitively disentangles the mixed clean and adversarial distribution (**data heterogeneity**) into two splits, guaranteeing the normalization statistics and re-scaling are exclusively performed in either domain. The loss function of AdvProp can be formulated as:

$$L_{\text{total}} = L(f_{\text{clean}}(x_{\text{clean}}), y) + L(f_{\text{adv}}(x_{\text{adv}}), y), \quad (7)$$

where  $f$  denotes the model and  $x, y$  denotes the input data, label respectively.  $L$  is the cross entropy loss. Therefore,  $f_{\text{clean}}$  denotes the model is using the clean branch BN.  $x_{\text{adv}}$  is the corresponding adversarial mini-batch generated by the model using adversarial branch BN. For simplicity, we call the two loss terms in Eq. 7 as **clean loss** and **adversarial loss** respectively.

However, one thing missed is that the domains of clean and adversarial images overlap largely, as adversarial images are generated by perturbing clean counterparts minimally. This inspires us to present a novel **SaAuxBN**, by leveraging domain-specific normalization and affine layers, and also a shared sandwich affine layer for homogeneity preserving. SaAuxBN can be defined as:

$$\mathbf{h} = \gamma_i (\gamma_{sa} \left( \frac{\mathbf{x} - \mu_i(\mathbf{x})}{\sigma_i(\mathbf{x})} \right) + \beta_{sa}) + \beta_i, i = 0, 1. \quad (8)$$

$\mu_i(\mathbf{x})$  and  $\sigma_i(\mathbf{x})$  denote the  $i$ -th (moving) mean and variance of input, where  $i = 0$  for adversarial images and  $i = 1$  for clean images. We use independent normalization layer to decouple the data from two different distributions, i.e., the clean and adversarial.

We replace AuxBN with SaAuxBN in AdvProp [61] and find it can further improve SA of the network with its clean branch. The experiments are conducted on CIFAR-10 [37] with ResNet-18 [26]. For a fair comparison, we follow the settings in [44]. In the adversarial training, we adopt  $\ell_\infty$  based 10 steps Projected Gradient Descent (PGD) [44] with step size  $\alpha = \frac{2}{255}$  and maximum perturbation magnitude

Table 3. Model performance (SA) using clean branch.

Evaluation	BN	ModeNorm	AuxBN (clean branch)	SaAuxBN (clean branch)
Clean (SA)	84.84	83.28	94.47	<b>94.62</b>

Table 4. Performance (SA&RA) using the adversarial branch.

Evaluation	BN	ModeNorm	AuxBN (adv branch)	SaAuxBN (adv branch)
Clean (SA)	<b>84.84</b>	83.28	83.42	84.08
PGD-10 (RA)	41.57	43.56	43.05	<b>44.93</b>
PGD-20 (RA)	40.02	41.85	41.60	<b>43.14</b>

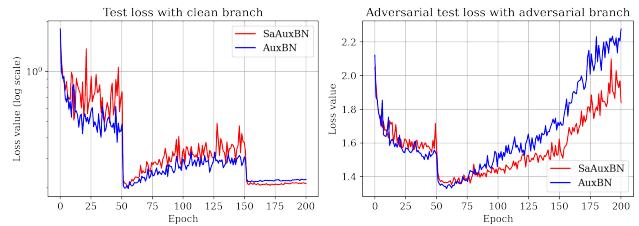


Figure 6. The **clean loss**  $L(f_{\text{clean}}(x_{\text{clean}}), y)$  and **adversarial loss**  $L(f_{\text{adv}}(x_{\text{adv}}), y)$  on testing set.

$\epsilon = \frac{8}{255}$ ; As for assessing RA, PGD-20 with the same configuration is adopted. The results are presented in Tab. 3.

We further conduct an experiment to test the Standard Testing Accuracy (SA) and Robust Testing Accuracy (RA) of the network using the adversarial branch of AuxBN and SaAuxBN. The comparison results are presented in Tab. 4. We can see that BN still achieves the highest performance on SA, but falls a lot on RA compared with other methods. Our proposed SaAuxBN is on par with the vanilla BN in terms of SA, while has significantly better results on RA than any other approaches. Compared with SaAuxBN, AuxBN suffers from both worse SA and RA.

We also visualize the testing clean loss and adversarial loss of models with AuxBN and SaAuxBN in Fig. 6, showing that the latter achieves lower value on both.

We additionally include ModeNorm [13] as an ablation in our experiments, which was proposed to deal with multi-modal distributions inputs, i.e., data heterogeneity. It shares some similarity with AuxBN as both consider multiple independent norms. ModeNorm achieves fair performance on both SA and RA, while still lower than SaAuxBN. The reason might be the output of ModeNorm is a summation of two features weighted by a set of learned gating functions, which still mixes the statistics from two domains, leading to inferior performance.

### 5.2. Arbitrary Style Transfer with Sandwich Adaptive Instance Normalization (SaAdaIN)

Huang & Belongie [28] achieves arbitrary style transfer by introducing Adaptive Instance Norm (AdaIN), which is an effective module to encode style information into feature space. The AdaIN framework is composed of three parts: Encoder, AdaIN, and Decoder. Firstly, the Encoder will ex-

tract content features and style features from content and style images. Then, the AdaIN is leveraged to perform style transfer on feature space, producing a stylized content feature. The Decoder is learned to decode the stylized content feature to stylized images. This framework is trained end-to-end with two loss terms, a content loss and a style loss. Concretely, AdaIN firstly performs a normalization on the content feature, then re-scale the normalized content feature with style feature’s statistic. It can be formulated as:

$$\mathbf{h} = \sigma(\mathbf{y})\left(\frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) + \mu(\mathbf{y}), \quad (9)$$

where  $\mathbf{y}$  is the style input,  $\mathbf{x}$  is the content input. Note that  $\mu$  and  $\sigma$  here are quite different from BN, which are performed along the spatial axes ( $H, W$ ) for each sample and each channel. The goal of style transfer is to extract the style information from the style input and render it to the content input. Obviously, style-dependent re-scale may be too loose and might further amplify the intrinsic **data heterogeneity** brought by the variety of the input content images, undermining the network’s ability of maintaining the content information in the output. In order to reduce the **data heterogeneity**, we propose to insert a shared sandwich affine layer after the normalization, which introduce **homogeneity** for the style-dependent re-scaling transformation. Hereby, we present **SaAdaIN**:

$$\mathbf{h} = \sigma(\mathbf{y})\left(\gamma_{sa}\left(\frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) + \beta_{sa}\right) + \mu(\mathbf{y}), \quad (10)$$

Besides AdaIN, we also include Instance-Level Meta Normalization with Instance Norm (ILM+IN) proposed by [31] as a task-specific comparison baseline. Its style-independent affine is not only conditioned on style information but also controlled by the input feature.

Our training settings for all models are kept identical with [28]. The network is trained with style loss and content loss. We use the training set of MS-COCO [42] and WikiArt [49] as the training content and style images dataset, and the validation set of MS-COCO and testing set of WikiArt are used as our validation set.

We depict the loss curves of the training phase in Fig. 7 (a). We can notice that both the content loss and style loss of the proposed SaAdaIN are lower than that of AdaIN and ILM+IN. This observation implies that the inserted sandwich affine layer makes the optimization easier. In Fig. 7 (b), we show the content and style loss on validation set. In contrast with the AdaIN model, the network with SaAdaIN achieves lower validation content and style loss, indicating the inserted sandwich affine layer also benefits the model’s generalizability. The visual results of style transfer are shown in Fig. 8. Compared with AdaIN and ILM-IN, SaBN generates more visual-appealing images.

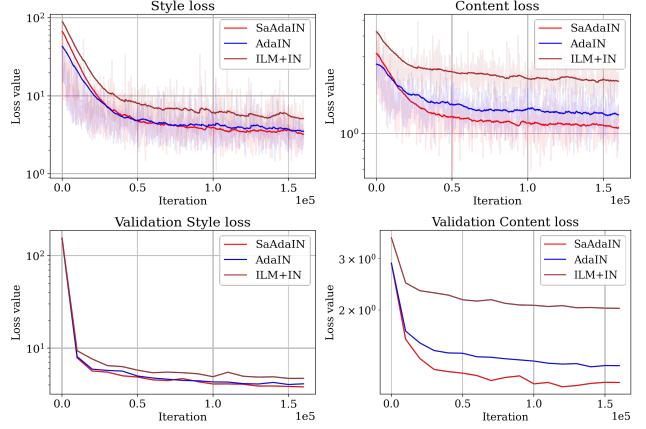


Figure 7. The content loss and the style loss of using AdaIN, ILM+IN and SaAdaIN on training and validation set. In the first row, the noisy shallow-color curves are the original data, and the foreground smoothed curves are obtained via applying exponential moving average on the original data.



Figure 8. The visual results of style transfer. The images on the top-left corner of each row are the reference style image. An ideally stylized output should be semantically similar to the content image, while naturally incorporate the style information from the referenced style image.

## 6. Conclusion

We present SaBN and its variants as plug-and-play normalization modules, which are motivated by addressing model & data heterogeneity issues. We demonstrate their effectiveness on several tasks, including neural architecture search, adversarial robustness, conditional image generation, and arbitrary style transfer. Our future work plans to investigate the performance of SaBN on more applications, such as semi-supervised learning [69].

## 7. Acknowledgment

Z. Wang is in part supported by IoBT REIGN subaward # 088831-18415, US Army Research Laboratory.

## References

- [1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
- [2] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. *arXiv preprint arXiv:1812.03981*, 2018.
- [3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [4] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.
- [5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.
- [6] Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. Distilling portable generative adversarial networks for image translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3585–3592, 2020.
- [7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2021.
- [8] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.
- [9] Xinyuan Chen, Chang Xu, Xiaokang Yang, Li Song, and Dacheng Tao. Gated-gan: Adversarial gated networks for multi-collection style transfer. *IEEE Transactions on Image Processing*, 28(2):546–560, 2018.
- [10] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019.
- [11] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Theoretical understanding of batch-normalization: A markov chain perspective. *arXiv preprint arXiv:2003.01652*, 2020.
- [12] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017.
- [13] Lucas Deecke, Iain Murray, and Hakan Bilen. Mode normalization. *arXiv preprint arXiv:1810.05466*, 2018.
- [14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [15] Guneeet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.
- [16] Minjing Dong, Yanxi Li, Yunhe Wang, and Chang Xu. Adversarially robust neural architectures. *arXiv preprint arXiv:2009.00902*, 2020.
- [17] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.
- [18] Xuanyi Dong and Yi Yang. Nas-bench-102: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.
- [19] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.
- [20] Jonathan Frankle, David J Schwab, and Ari S Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *arXiv preprint arXiv:2003.00152*, 2020.
- [21] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.
- [22] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3224–3234, 2019.
- [23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [25] Ishaaq Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [27] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.
- [28] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.
- [29] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in neural information processing systems*, pages 1945–1953, 2017.
- [30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

- [31] Songhao Jia, Ding-Jie Chen, and Hwann-Tzong Chen. Instance-level meta normalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4865–4873, 2019.
- [32] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.
- [33] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [34] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.
- [35] Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Ming Zhou, Klaus Neymeyr, and Thomas Hofmann. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. *arXiv preprint arXiv:1805.10694*, 2018.
- [36] Zhi Kou, Kaichao You, Mingsheng Long, and Jianmin Wang. Stochastic normalization. *Advances in Neural Information Processing Systems*, 33, 2020.
- [37] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.
- [38] Boyi Li, Felix Wu, Kilian Q Weinberger, and Serge Belongie. Positional normalization. In *Advances in Neural Information Processing Systems*, pages 1622–1634, 2019.
- [39] Xilai Li, Wei Sun, and Tianfu Wu. Attentive normalization. *arXiv preprint arXiv:1908.01259*, 2019.
- [40] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.
- [41] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.
- [42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [43] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
- [44] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [45] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.
- [46] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147, 2017.
- [47] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.
- [48] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.
- [49] Kiri Nichol. Painter by numbers, wikiart, 2016.
- [50] Nicolas Papernot and Patrick McDaniel. Extending defensive distillation. *arXiv preprint arXiv:1705.05264*, 2017.
- [51] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.
- [52] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.
- [53] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Rethinking normalization and elimination singularity in neural networks. *arXiv preprint arXiv:1911.09738*, 2019.
- [54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.
- [55] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.
- [56] Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11237–11246, 2020.
- [57] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- [58] Yi Wang, Ying-Cong Chen, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Attentive normalization for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5094–5103, 2020.
- [59] Junru Wu, Xiyang Dai, Dongdong Chen, Yinpeng Chen, Mengchen Liu, Ye Yu, Zhangyang Wang, Zicheng Liu, Mei Chen, and Lu Yuan. Weak nas predictors are all you need. *arXiv preprint arXiv:2102.10490*, 2021.
- [60] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.
- [61] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, and Quoc V Le. Adversarial examples improve image recognition. *arXiv preprint arXiv:1911.09665*, 2019.
- [62] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training. *arXiv preprint arXiv:1906.03787*, 2019.
- [63] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.
- [64] Yixing Xu, Yunhe Wang, Kai Han, Yehui Tang, Shangling Jui, Chunjing Xu, and Chang Xu. Renas: Relativistic evaluation of neural architecture search. In *Proceedings of*

*the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4411–4420, 2021.

- [65] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Jianyuan Guo, Wei Zhang, Chao Xu, Chunjing Xu, Dacheng Tao, and Chang Xu. Hournas: Extremely fast neural architecture search through an hourglass lens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10896–10906, 2021.
- [66] Chris Ying, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. *arXiv preprint arXiv:1902.09635*, 2019.
- [67] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.
- [68] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.
- [69] Michał Zajęc, Konrad Żołna, and Stanisław Jastrzębski. Split batch normalization: Improving semi-supervised learning under domain shift. *arXiv preprint arXiv:1904.03515*, 2019.
- [70] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *arXiv preprint arXiv:2001.10422*, 2020.
- [71] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.
- [72] Jie Zhang, Dongdong Chen, Jing Liao, Weiming Zhang, Gang Hua, and Nenghai Yu. Passport-aware normalization for deep model protection. *Advances in Neural Information Processing Systems*, 33, 2020.
- [73] Heliang Zheng, Jianlong Fu, Yanhong Zeng, Jiebo Luo, and Zheng-Jun Zha. Learning semantic-aware normalization for generative adversarial networks. *Advances in Neural Information Processing Systems*, 33, 2020.
- [74] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.